Plantuml Mode Basic

Table of Contents

- 1. 使用 PlantUML画UML
  - 1.1. 时序图 (信息交流)
    - 1.1.1. 示例
    - 1.1.2. 声明参与者
      - 1.1.2.1. 普通示例
      - 1.1.2.2. 样式示例
      - 1.1.2.3. 参与者的显示顺序
      - 1.1.2.4. 参与者使用非字母内容
      - 1.1.2.5. 给自己发送消息
      - 1.1.2.6. 文本对齐
      - 1.1.2.7. 改变箭头形状
      - 1.1.2.8. 改变箭头颜色
      - 1.1.2.9. 消息序列编号
        - 1.1.2.9.1. 自动编号
        - 1.1.2.9.2. 自定义编号
        - 1.1.2.9.3. 自定义HTML样式编号
        - 1.1.2.9.4. 暂停和恢复自动编号
      - 1.1.2.10. 页面头部 标题 页脚
      - 1.1.2.11. 切割图
      - 1.1.2.12. 组织信息
      - 1.1.2.13. 次要组标签
      - 1.1.2.14. 消息上的笔记
      - 1.1.2.15. 其他笔记的位置和颜色
      - 1.1.2.16. 改变笔记的形状 [六边形】[长方形]
      - 1.1.2.17. 笔记穿过所有参与者(跨过)
      - 1.1.2.18. 多个笔记在同一级别对齐
      - 1.1.2.19. Creole 和 HTML
      - 1.1.2.20. 分隔线或分隔符
      - 1.1.2.21. 引用
      - 1.1.2.22. 延迟
      - 1.1.2.23. 文字环绕
      - 1.1.2.24. 空格
    - 1.1.3. 生命线
      - 1.1.3.1. 生命线的激活和销毁
      - 1.1.3.2. 嵌套生命线
      - 1.1.3.3. 生命线的自动激活及 return 关键字
      - 1.1.3.4. return
      - 1.1.3.5. 参与者创建
      - 1.1.3.6. 激活、停用、创建的快捷方式语法
      - 1.1.3.7. 传入和传出消息 - 针对某个部分
        - 1.1.3.7.1. 局部图表语法
      - 1.1.3.8. 传入和传出消息的短箭头
      - 1.1.3.9. 锚点和持续时间
      - 1.1.3.10. 立体类型和斑点
      - 1.1.3.11. 有关标题的更多信息 (可以添加图片!)
      - 1.1.3.12. 参与者环绕方框
      - 1.1.3.13. 移除 Foot Boxes
      - 1.1.3.14. 皮肤参数

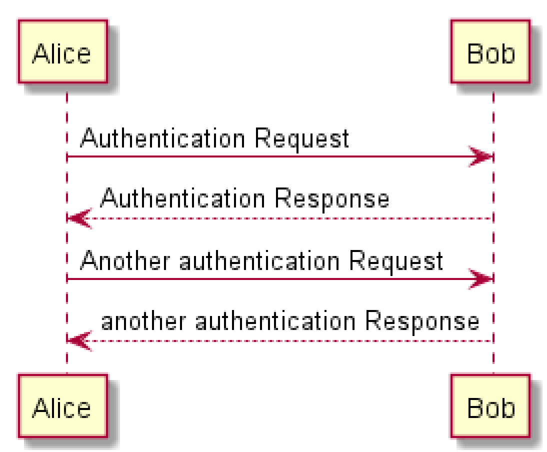
- 1.1.3.14.1. 案例二
- 1.1.3.15. 改变内间距
- 1.1.4. 各种箭头
  - 1.1.4.1. 正常的箭头
  - 1.1.4.2. 收到的信息 (带 [)
  - 1.1.4.3. 发出的信息 (带])
  - 1.1.4.4. 简短的传入信息（带"?"）
  - 1.1.4.5. 简短的发出信息（带"?"）
- 1.1.5. 皮肤
  - 1.1.5.1. 默认皮肤
  - 1.1.5.2. 生命线 strategy
  - 1.1.5.3. 严格控制样式
  - 1.1.5.4. 隐藏未链接的参与者
- 1.1.6. FAQ
  - 1.1.6.1. uml 中的生命线

1 使用 PlantUML画UML

1.1 时序图 (信息交流)

1.1.1 示例

```
@startuml
/'
时序图示例
'/
Alice -> Bob: Authentication Request
Bob --> Alice: Authentication Response
Alice -> Bob: Another authentication Request
Alice <-- Bob: another authentication Response
@enduml
```



1.1.2 声明参与者

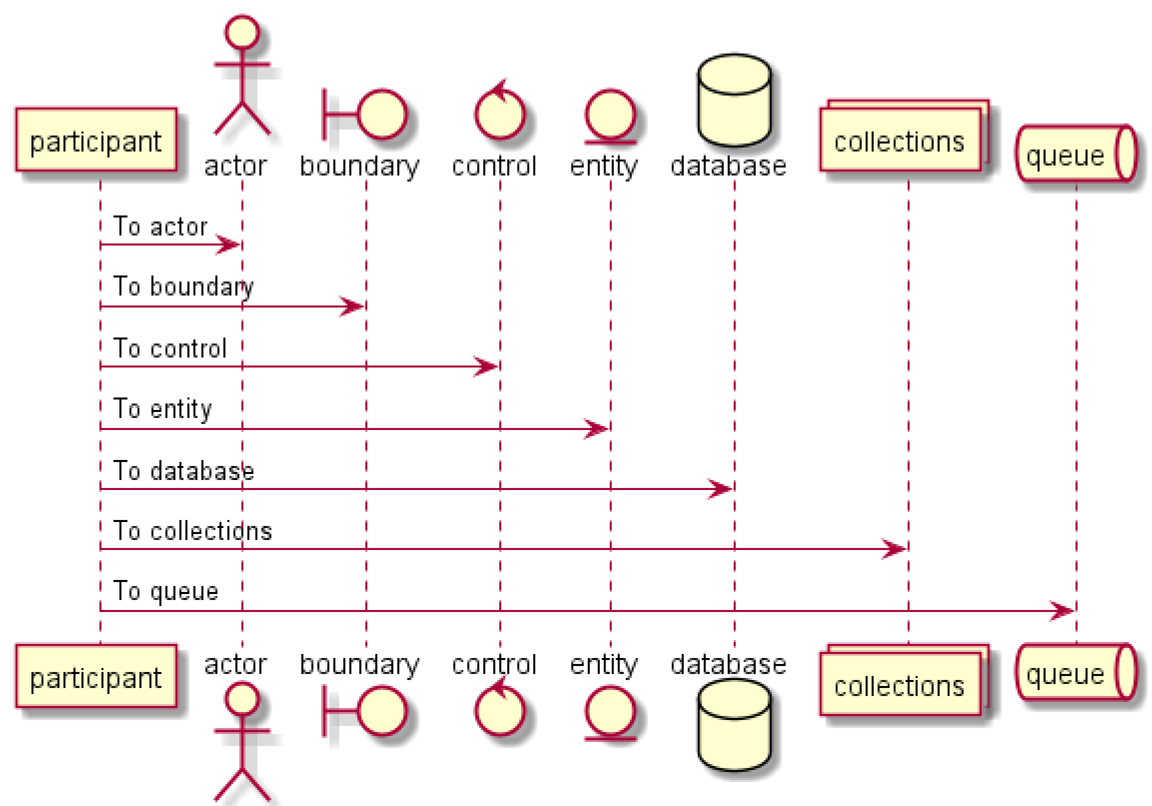
目的是控制参与者,体现的是让它们干什么.

- 顺序性
- 形状

1.1.2.1 普通示例

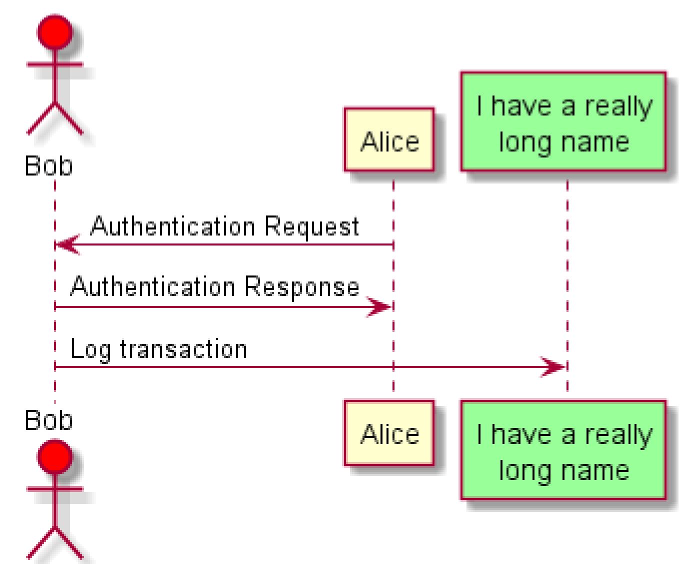
```
@startuml
' 声明参与者,使用as来重命名参与者
participant participant as Foo
actor actor as Foo1
boundary boundary as Foo2
control control as Foo3
entity entity as Foo4
database database as Foo5
collections collections as Foo6
queue queue as Foo7

' 控制参与者 - 指向谁
Foo -> Foo1 : To actor
Foo -> Foo2 : To boundary
Foo -> Foo3 : To control
Foo -> Foo4 : To entity
Foo -> Foo5 : To database
Foo -> Foo6 : To collections
Foo -> Foo7: To queue
@enduml
```



1.1.2.2 样式示例

```
@startuml
actor Bob #red
participant Alice
participant "I have a really\nlong name" as L #99FF99
/' You can also declare: participant L as "I have a really\nlong name" #99FF99 '/
Alice->Bob: Authentication Request
Bob->Alice: Authentication Response
Bob->L: Log transaction
@enduml
```

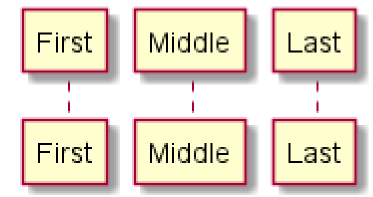



```
actor Bob #red
participant Alice
participant "I have a really\nlong name" as L #99FF99
' You can also declare: participant L as "I have a really\nlong name" #99FF99 '
```

```
Alice->Bob: Authentication Request
Bob->Alice: Authentication Response
Bob->L: Log transaction
```

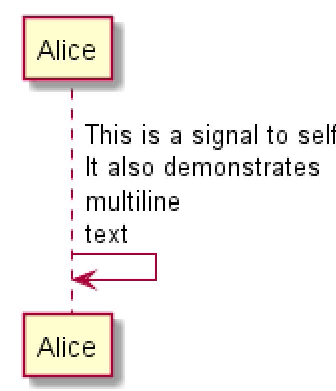
1.1.2.3 参与者的显示顺序

```
@startuml
participant Last order 30
participant Middle order 20
participant First order 10
@enduml
```



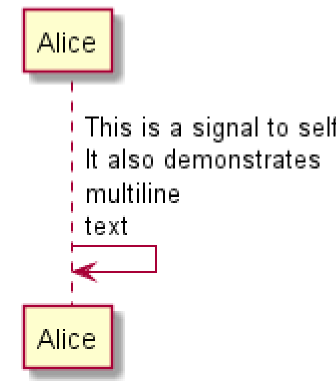
1.1.2.4 参与者使用非字母内容

```
@startuml
Alice -> "Bob()" : Hello
"Bob()" -> "This is very\nlong" as Long
' You can also declare:
' "Bob()" -> Long as "This is very\nlong"
Long --> "Bob()" : ok
@enduml
```



1.1.2.5 给自己发送消息

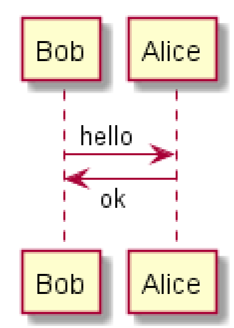
```
@startuml
Alice->Alice: This is a signal to self.\nIt also demonstrates\nmultiline \ntext
@enduml
```



1.1.2.6 文本对齐

将相应信息文本放在箭头的下方

```
skinparam responseMessageBelowArrow true
Bob -> Alice : hello
Alice -> Bob : ok
```

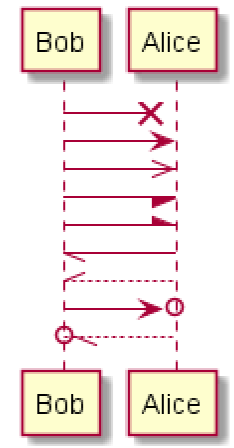


1.1.2.7 改变箭头形状

- 丢失的消息
- 底部箭头
- 顶部箭头
- 虚线箭头
- 双向箭头

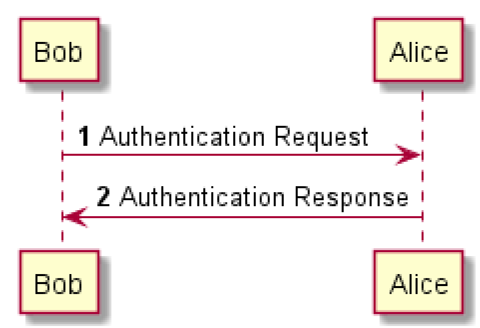
```
@startuml
Bob ->x Alice
Bob -> Alice
Bob ->> Alice
Bob -/ Alice
Bob -\ Alice
```

```
Bob \\- Alice
Bob //-- Alice
Bob ->o Alice
Bob o\\-- Alice
@enduml
```



1.1.2.8 改变箭头颜色

```
@startuml
Bob -[#red]> Alice : hello
Alice -[#0000FF]->Bob : ok
@enduml
```

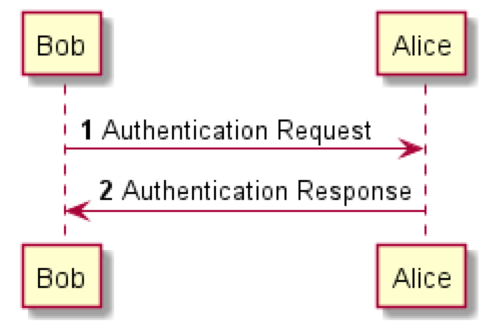


1.1.2.9 消息序列编号

1.1.2.9.1 自动编号

```
@startuml
autonumber
```

```
Bob -> Alice : Authentication Request
Bob <- Alice : Authentication Response
@enduml
```

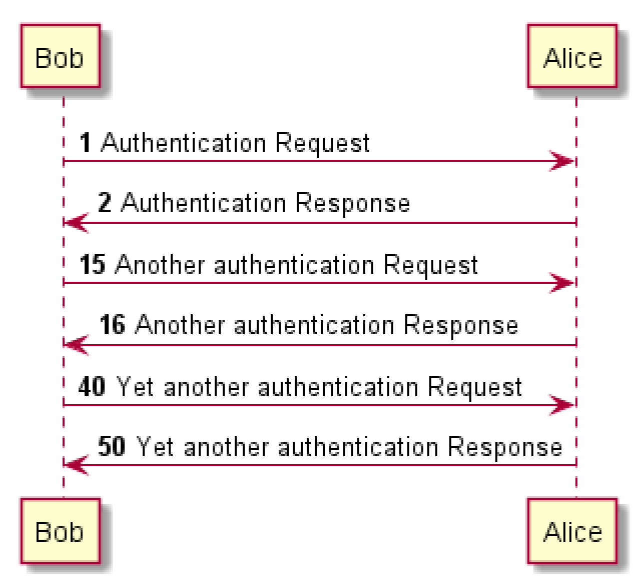


1.1.2.9.2 自定义编号

```
@startuml
autonumber
Bob -> Alice : Authentication Request
Bob <- Alice : Authentication Response

autonumber 15
Bob -> Alice : Another authentication Request
Bob <- Alice : Another authentication Response

autonumber 40 10
Bob -> Alice : Yet another authentication Request
Bob <- Alice : Yet another authentication Response
@enduml
```

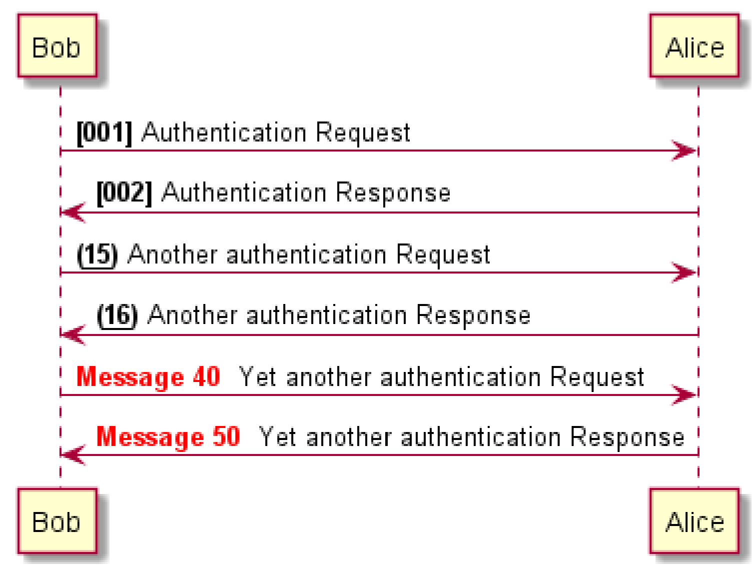


1.1.2.9.3 自定义HTML样式编号

```
@startuml
autonumber "<b>[000]"
Bob -> Alice : Authentication Request
Bob <- Alice : Authentication Response

autonumber 15 "<b>(<u>##</u>)"
Bob -> Alice : Another authentication Request
Bob <- Alice : Another authentication Response

autonumber 40 10 "<font color=red><b>Message 0 "
Bob -> Alice : Yet another authentication Request
Bob <- Alice : Yet another authentication Response
@enduml
```



1.1.2.9.4 暂停和恢复自动编号

也可以使用 autonumber stop 和 autonumber resume `//increment//` `//format//` 分别暂停和恢复自动编号.

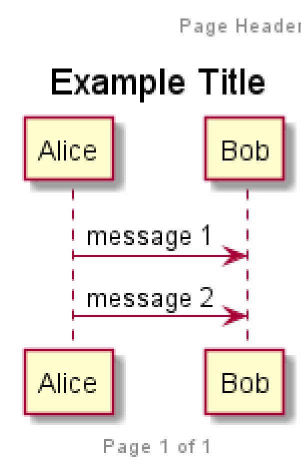
```
@startuml
autonumber 10 10 "<b>[000]"
Bob -> Alice : Authentication Request
Bob <- Alice : Authentication Response

autonumber stop
Bob -> Alice : dummy
```

```
autonumber resume "<font color=red><b>Message 0 "
Bob -> Alice : Yet another authentication Request
Bob <- Alice : Yet another authentication Response

autonumber stop
Bob -> Alice : dummy

autonumber resume 1 "<font color=blue><b>Message 0 "
Bob -> Alice : Yet another authentication Request
Bob <- Alice : Yet another authentication Response
@enduml
```

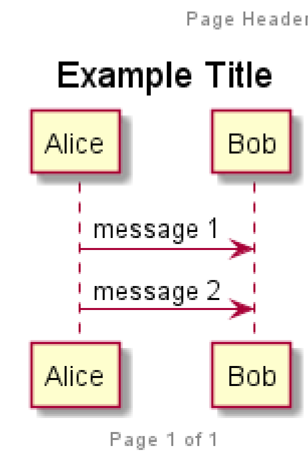


1.1.2.10 页面头部 标题 页脚

```
@startuml
header Page Header
footer Page %page% of %lastpage%

title Example Title

Alice -> Bob : message 1
Alice -> Bob : message 2
@enduml
```



1.1.2.11 切割图

newpage 关键字用于将图表拆分为多个图像

您可以在 newpage 关键字之后为新页面添加标题。如果有的话，这个标题会覆盖之前指定的标题。

这对于 Word 在多页上打印长图非常方便.

尽管它如下所示,但打印的时候的确显示全部.

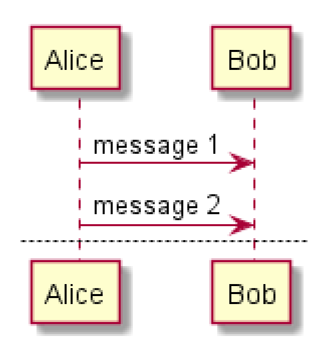
```
@startuml
Alice -> Bob : message 1
Alice -> Bob : message 2

newpage

Alice -> Bob : message 3
Alice -> Bob : message 4

newpage A title for the\nlast page

Alice -> Bob : message 5
Alice -> Bob : message 6
@enduml
```



1.1.2.12 组织信息

下面的关键词可以用来组织信息

- alt/else
- opt
- loop
- par
- break
- critical
- group

添加的文本可能会显示在头部,end 关键词用于关闭 group.

```
@startuml

Alice -> Bob: Authentication Request

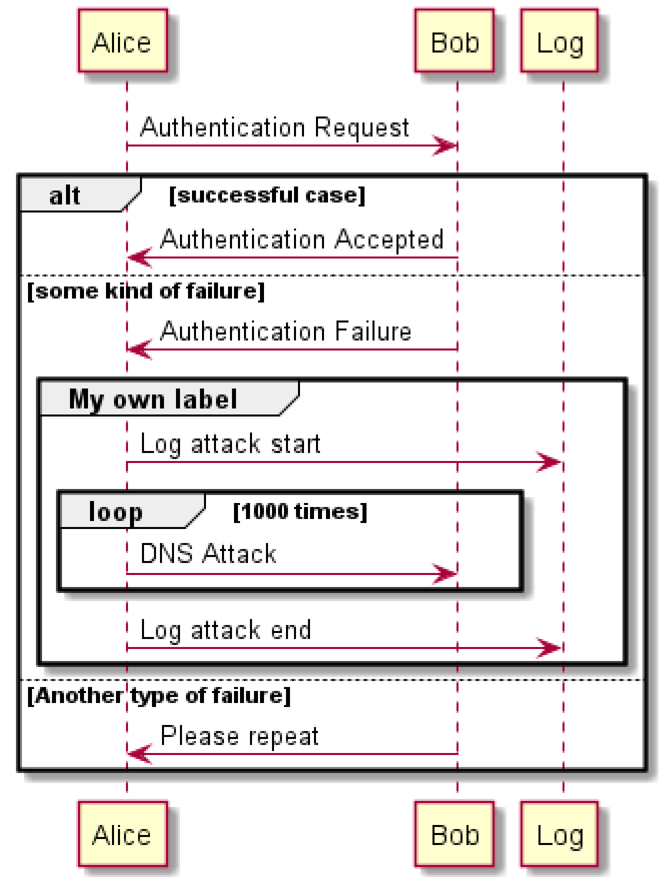
alt successful case
        Bob -> Alice: Authentication Accepted

else some kind of failure

        Bob -> Alice: Authentication Failure
        group My own label

        Alice -> Log : Log attack start
        loop 1000 times
        Alice -> Bob: DNS Attack
        end
        Alice -> Log : Log attack end
        end

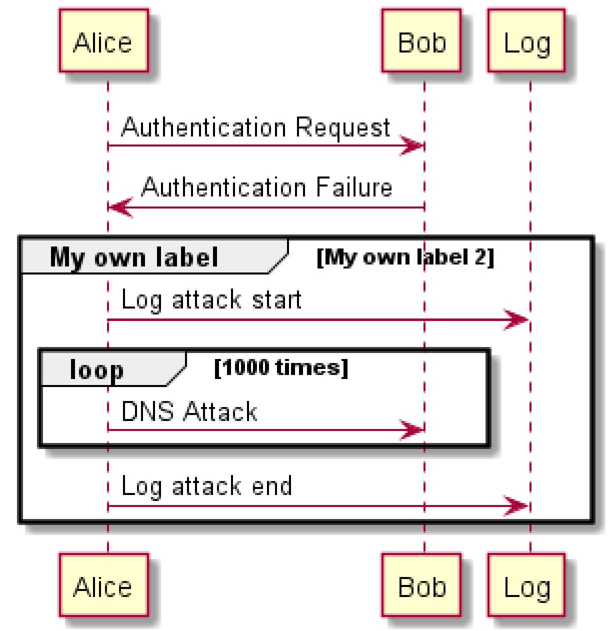
else Another type of failure
        Bob -> Alice: Please repeat
end
@enduml
```



1.1.2.13 次要组标签

```
@startuml
Alice -> Bob: Authentication Request
Bob -> Alice: Authentication Failure

group My own label [My own label 2]
        Alice -> Log : Log attack start
        loop 1000 times
        Alice -> Bob: DNS Attack
        end
        Alice -> Log : Log attack end
end
@enduml
```

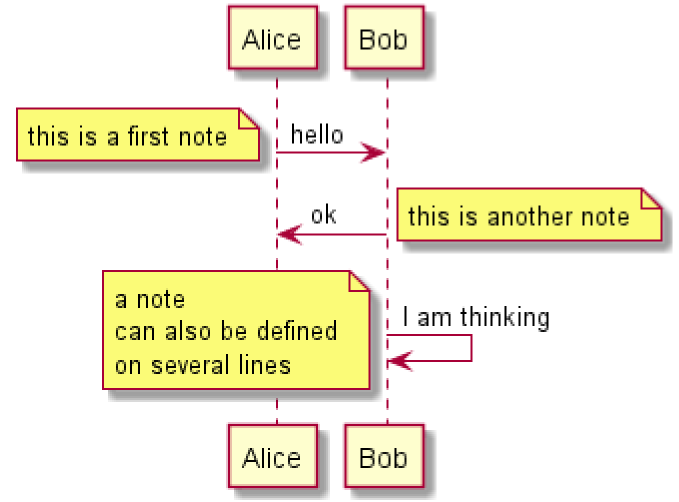


1.1.2.14 消息上的笔记

```
@startuml
Alice->Bob : hello
note left: this is a first note

Bob->Alice : ok
note right: this is another note

Bob->Bob : I am thinking
note left
        a note
        can also be defined
        on several lines
end note
@enduml
```



1.1.2.15 其他笔记的位置和颜色

可以放置与参与者相关的笔记,

```
@startuml

participant Alice
participant Bob

note left of Alice #aqua
This is displayed
left of Alice.
end note

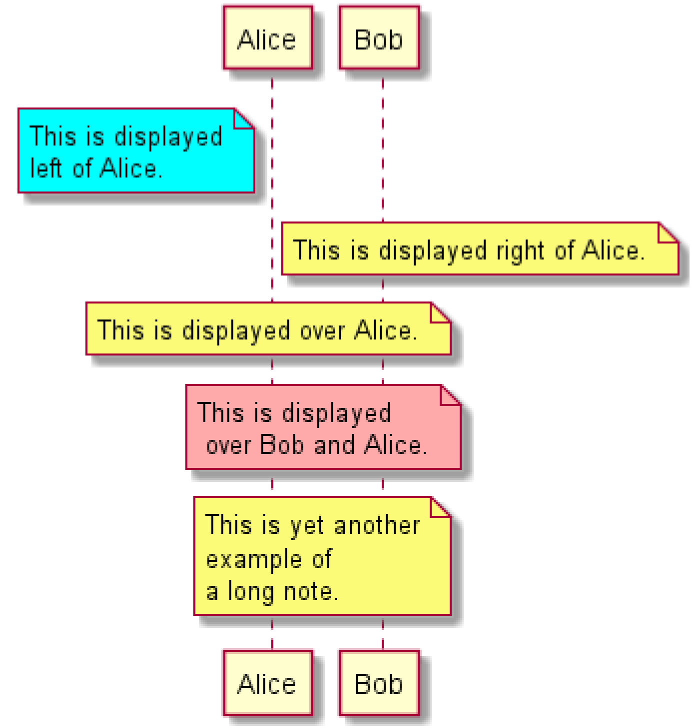
note right of Alice: This is displayed right of Alice.

note over Alice: This is displayed over Alice.

note over Alice, Bob #FFAAAA: This is displayed\n over Bob and Alice.

note over Bob, Alice
This is yet another
example of
a long note.
end note

@enduml
```



1.1.2.16 改变笔记的形状【六边形】【长方形】

```
@startuml

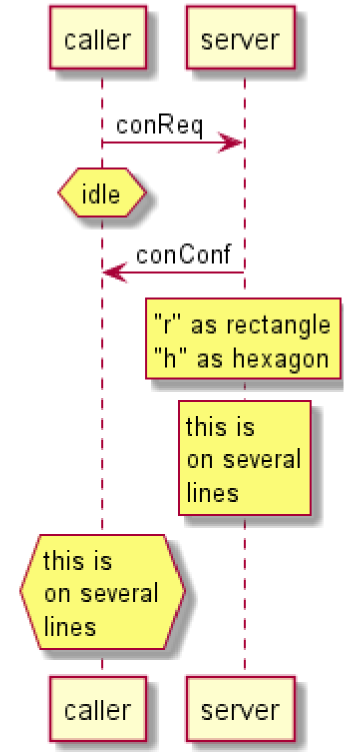
caller -> server : conReq
hnote over caller : idle

caller <- server : conConf
rnote over server
        "r" as rectangle
        "h" as hexagon
endrnote

rnote over server
this is
on several
lines
endrnote

hnote over caller
this is
on several
lines
endhnote

@enduml
```



1.1.2.17 笔记穿过所有参与者[跨过]

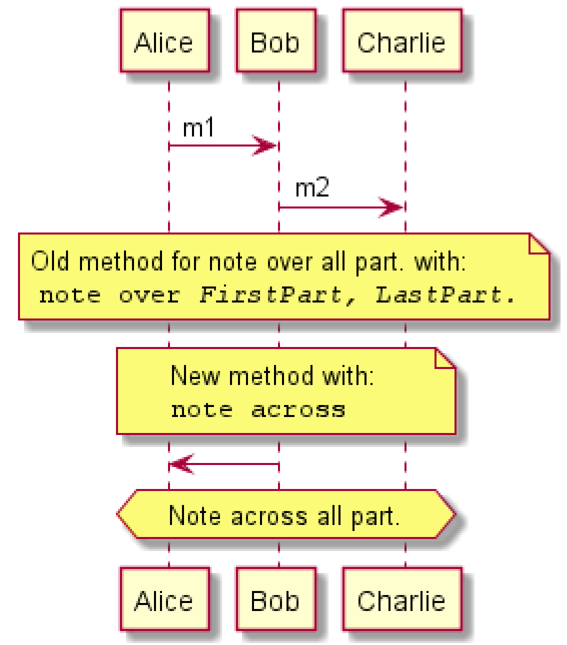
```
@startuml

Alice->Bob:m1
Bob->Charlie:m2

note over Alice, Charlie: Old method for note over all part. with:\n ""note over //FirstPart
note across: New method with:\n""note across""

Bob->Alice
hnote across:Note across all part.

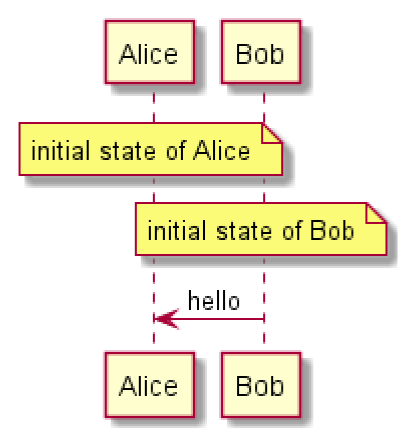
@enduml
```



1.1.2.18 多个笔记在同一级别对齐

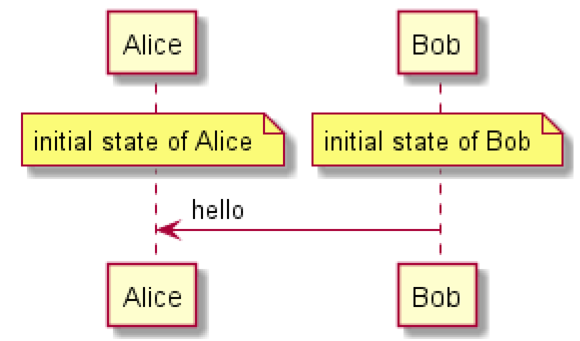
默认情况下笔记不对齐

```
@startuml
note over Alice : initial state of Alice
note over Bob : initial state of Bob
Bob -> Alice : hello
@enduml
```



使用/让其对齐.

```
@startuml
note over Alice : initial state of Alice
/ note over Bob : initial state of Bob
Bob -> Alice : hello
@enduml
```



1.1.2.19 Creole 和 HTML

```
@startuml
participant Alice
participant "The **Famous** Bob" as Bob

Alice -> Bob : hello --there--
... Some ~~long delay~~ ...
Bob -> Alice : ok

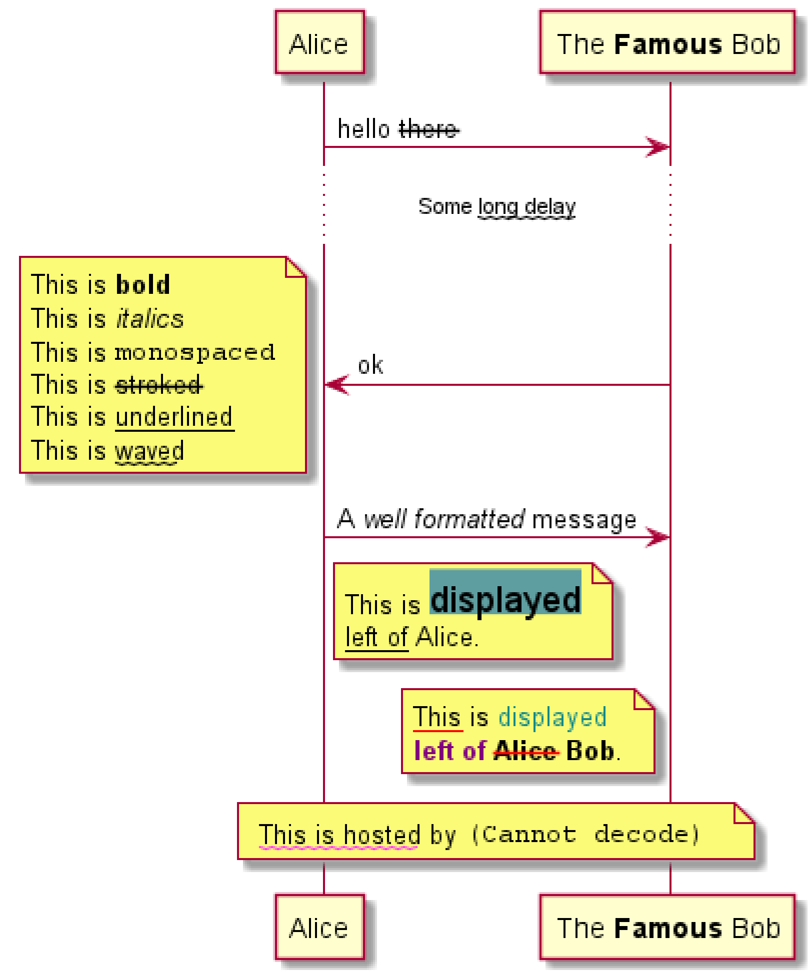
note left
This is **bold**
This is //italics//
This is ""monospaced""
This is --stroked--
This is __underlined__
This is ~~waved~~
end note

Alice -> Bob : A //well formatted// message
note right of Alice
This is <back:cadetblue><size:18>displayed</size></back>
__left of__ Alice.
end note

note left of Bob
<u:red>This</u> is <color #118888>displayed</color>
**<color purple>left of</color> <s:red>Alice</strike> Bob**.
end note

note over Alice, Bob
<w:#FF33FF>This is hosted</w> by <img sourceforge.jpg>
end note
```

```
@enduml
```



1.1.2.20 分隔线或分隔符

```
@startuml

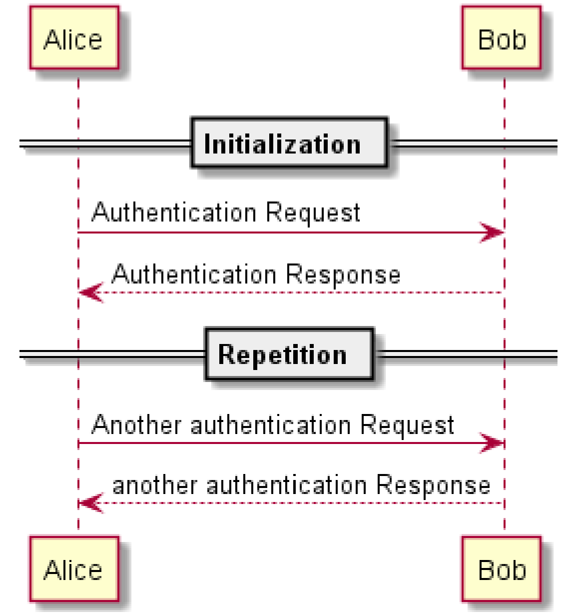
== Initialization ==

Alice -> Bob: Authentication Request
Bob --> Alice: Authentication Response

== Repetition ==

Alice -> Bob: Another authentication Request
Alice <-- Bob: another authentication Response

@enduml
```



1.1.2.21 引用

参与者Alice

行动者 Bob

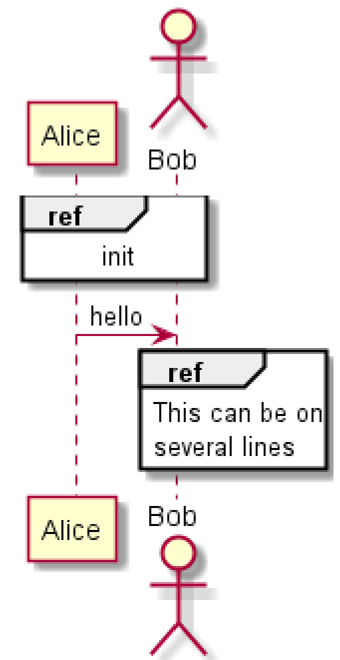
主要在图表中使用引用,ref over

```
@startuml
participant Alice
actor Bob

ref over Alice, Bob : init

Alice -> Bob : hello

ref over Bob
        This can be on
        several lines
end ref
@enduml
```



1.1.2.22 延迟

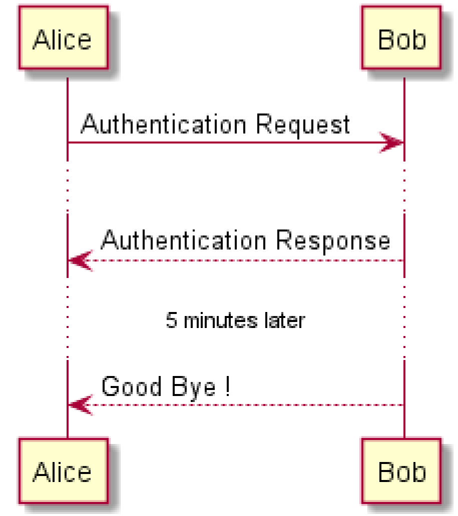
你可以使用 … 来表示图中的延迟.也可以使用延迟放置消息.

```
@startuml
Alice -> Bob: Authentication Request
...

Bob --> Alice: Authentication Response

...5 minutes later...

Bob --> Alice: Good Bye !
@enduml
```



1.1.2.23 文字环绕

要中断长消息,你可以手动添加文本。

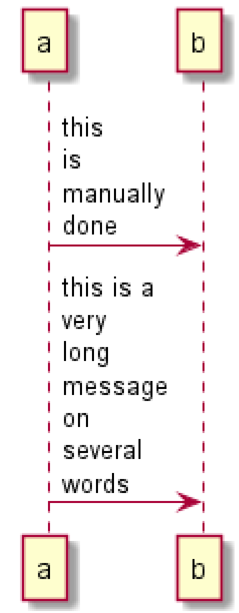
另一种选择是使用 maxMessageSize 设置:

```
@startuml
skinparam maxMessageSize 50

participant a
participant b

a -> b :this\nis\nmanually\ndone

a -> b :this is a very long message on several words
@enduml
```



1.1.2.24 空格

您可以使用 ||| 表示图中的一些间距.

也可以指定要使用的像素数

```
@startuml

Alice -> Bob: message 1
Bob --> Alice: ok

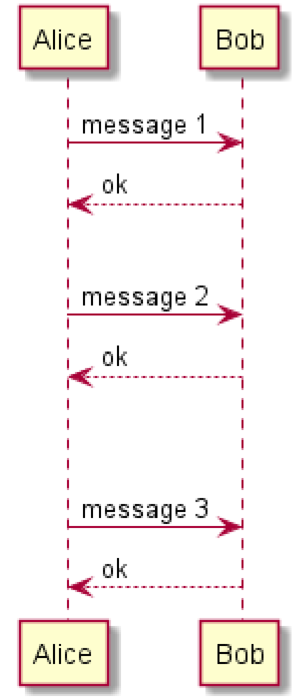
|||

Alice -> Bob: message 2
Bob --> Alice: ok

||45||

Alice -> Bob: message 3
Bob --> Alice: ok

@enduml
```



1.1.3 生命线

1.1.3.1 生命线的激活和销毁

激活和销毁都用于表示参与者的活动.

一旦参与者被激活,它的生命线就会出现。

激活和停用都适用于上一条消息。销毁表示参与者生命线的结束。

```
@startuml

participant User
User -> A: DoWork

' 参与者A的生命线被激活
activate A
A -> B: << createRequest >>

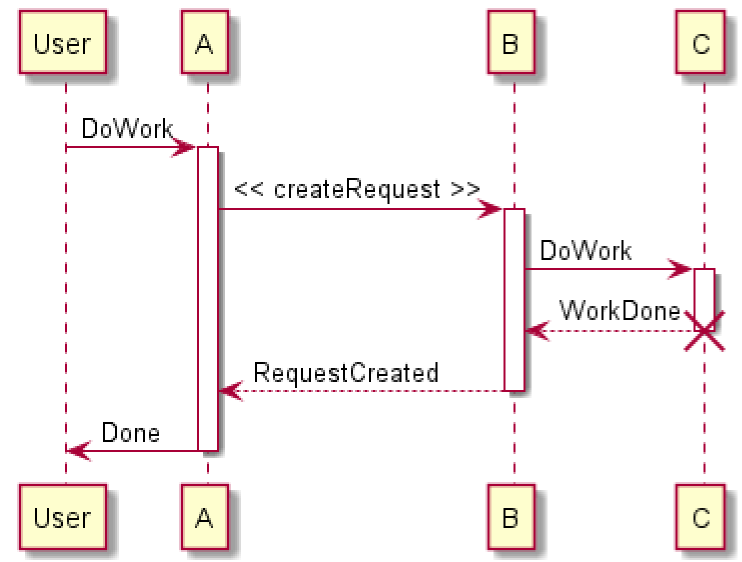
' 生命线 B
activate B
B -> C: DoWork

' 生命线 C
activate C
C --> B: WorkDone
destroy C                       /'销毁生命线 C'/

B --> A: RequestCreated
deactivate B                    /'停用 生命线 B'/
```

```
A -> User: Done
deactivate A                    /'停用生命线 A'/

@enduml
```



1.1.3.2 嵌套生命线

可以嵌套生命线,也能在生命线上添加颜色.

```
@startuml
participant User

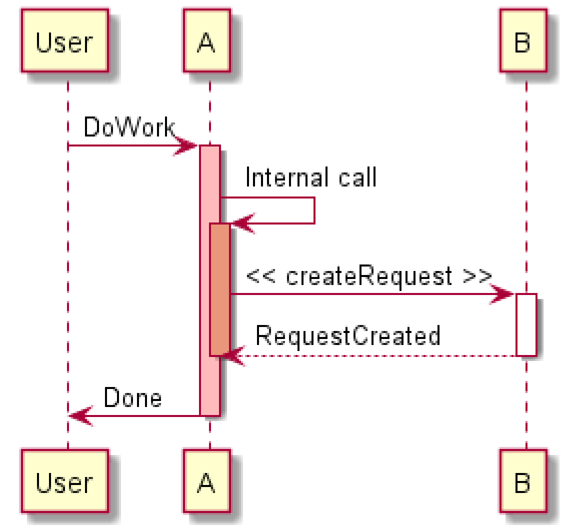
User -> A: DoWork
activate A #FFBBBB              /'激活生命线 A'/

A -> A: Internal call
activate A #DarkSalmon          /'内部调用,再次激活生命线 A'/

A -> B: << createRequest >>
activate B                      /'激活生命线 B'/

B --> A: RequestCreated
deactivate B                    /'暂停生命线 B,可以了解到 B 这条生命线是 生命线 A 中的嵌套内容'/
deactivate A                    /'暂停生命线 A'/

A -> User: Done
deactivate A                    /'暂停生命线 A'/
@enduml
```



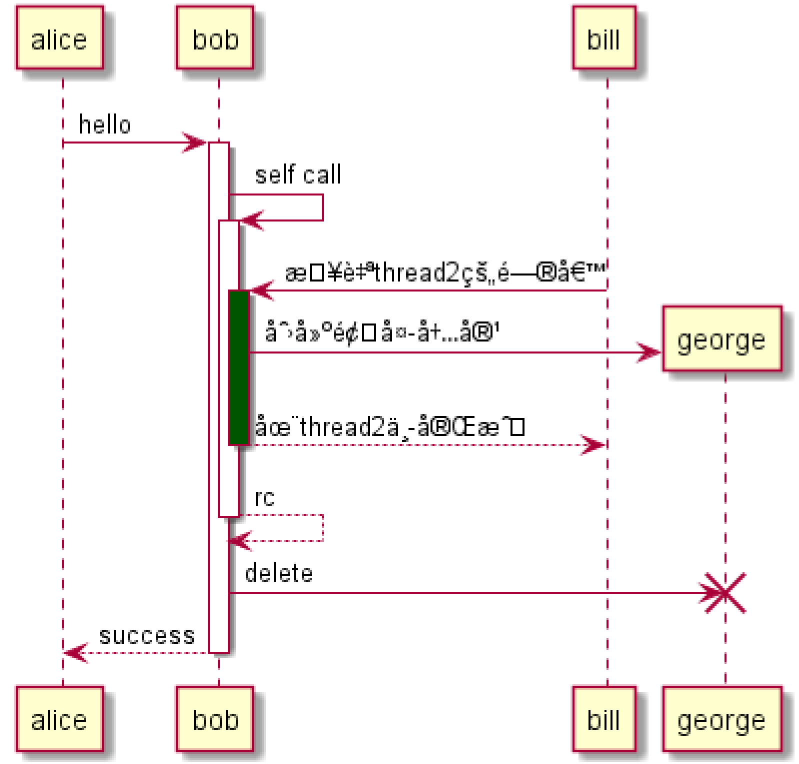
1.1.3.3 生命线的自动激活及 return 关键字

```
@startuml
autoactivate on

alice -> bob : hello

bob -> bob : self call
bill -> bob #005500 : 来自thread2的问候
bob -> george ** : 创建额外内容
return 在thread2中完成              /'生命线 bob 结束'/
return rc

bob -> george !! : delete
return success
@enduml
```



1.1.3.4 return

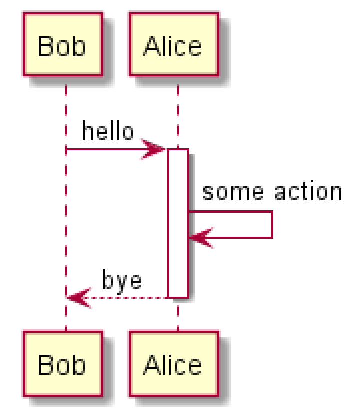
命令`return`生成一个携带可选文本标签的返回信息.

return point(返回点)是引起最近一次生命线激活的地方

语法是`return label`,如果提供的标签是常规信息可以接受的任何字符串.

```
@startuml
Bob -> Alice : hello
activate Alice

Alice -> Alice : some action
return bye
@enduml
```



1.1.3.5 参与者创建

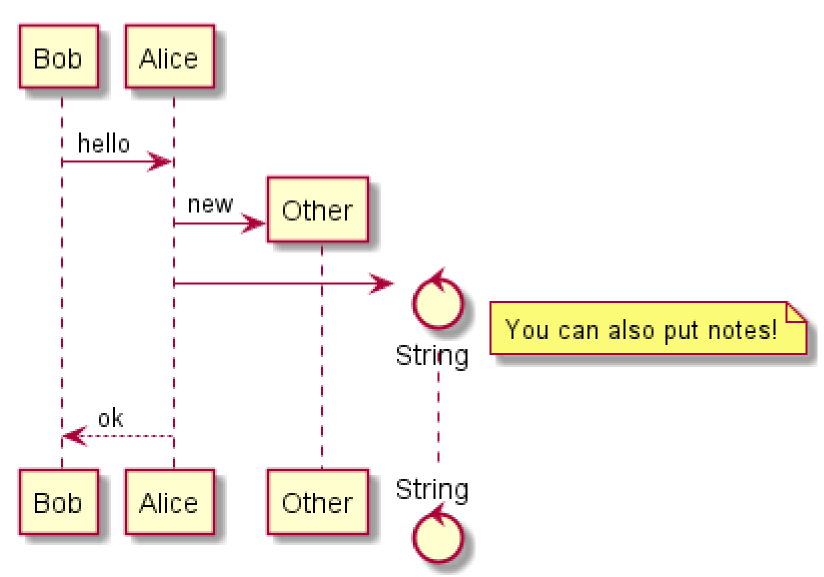
你可以在第一次接收消息之前使用create关键字,以强调这个消息实际上是在创建这个新对象.

```
@startuml
Bob -> Alice : hello

create Other
Alice -> Other : new

create control String
Alice -> String
note right : You can also put notes!

Alice --> Bob : ok
@enduml
```



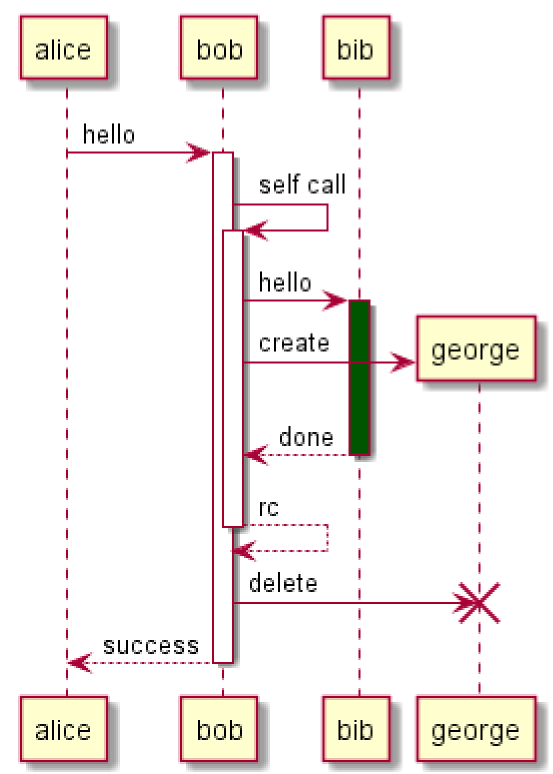
1.1.3.6 激活、停用、创建的快捷方式语法

在指定了目标参与者之后，可以立即使用以下语法.

- ++ 激活目标(可选择在这之后加一个#色).
- -- 停用来源
- ** 创建一个目标的实例
- !! 销毁目标的一个实例

```
@startuml
alice -> bob ++ : hello          /'激活参与交互的生命线'/
bob -> bob ++ : self call        /'激活自我嵌套交互'/
bob -> bib ++ #005500 : hello    /'激活新的交互'/
bob -> george ** : create        /'创建一个目标实例'/
return done                      /'返回信息'/
return rc                        /'返回信息'/

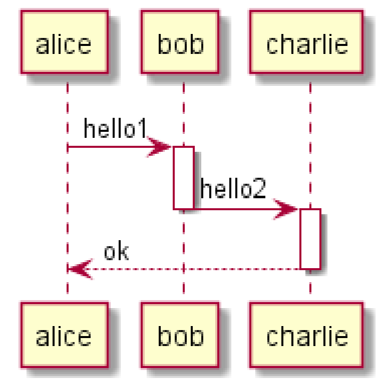
bob -> george !! : delete        /'销毁目标实例 '/
return success                   /'返回信息'/
@enduml
```



你可以在同一条线上混合激活和停用:

```
@startuml
alice -> bob ++ : hello1
bob -> charlie --++ : hello2
```

```
charlie --> alice -- : ok
@enduml
```

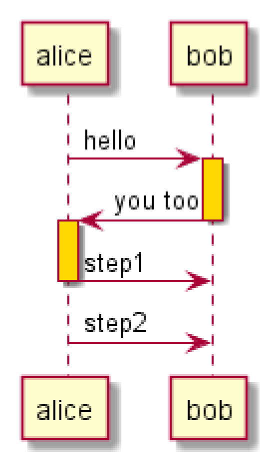



```
@startuml
@startuml

alice -> bob ++ #gold: hello
bob -> alice ++ #gold: you too

alice -> bob --: step1
alice -> bob : step2

@enduml
@enduml
```



1.1.3.7 传入和传出消息 - 针对某个部分

如果你想把注意力集中在图表的某个部分,你可以使用传入箭头或传出箭头.

使用方括号来表示图表的左边"["或右边"]".

```
@startuml
[-> A: DoWork

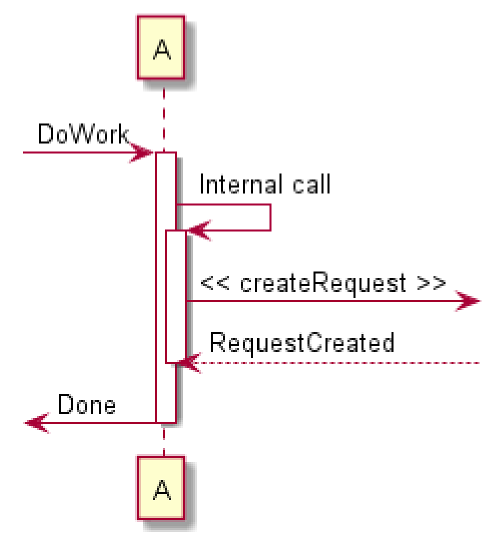
activate A

A -> A: Internal call
activate A

A ->] : << createRequest >>

A<--] : RequestCreated
deactivate A

[<- A: Done
deactivate A
@enduml
```



1.1.3.7.1 局部图表语法

```
@startuml
participant Alice
participant Bob #lightblue
Alice -> Bob
Bob -> Carol

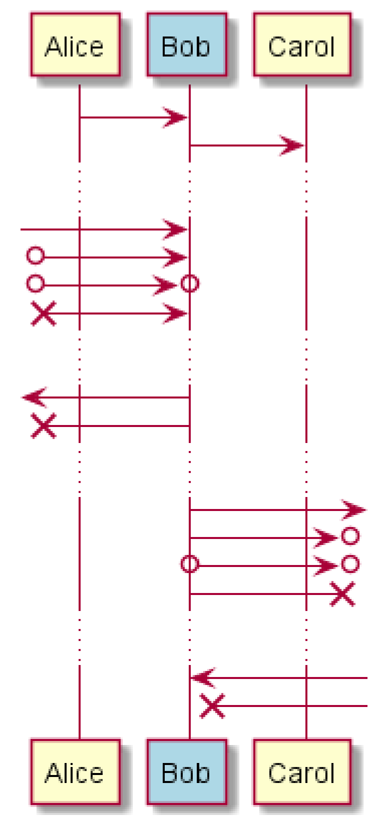
...
[-> Bob
[o-> Bob
[o->o Bob
[x-> Bob

...
```

```
[<- Bob
[x<- Bob

...
Bob ->]
Bob ->o]
Bob o->o]
Bob ->x]

...
Bob <-]
Bob x<-]
@enduml
```



1.1.3.8 传入和传出消息的短箭头

可以使用带?的短箭头.

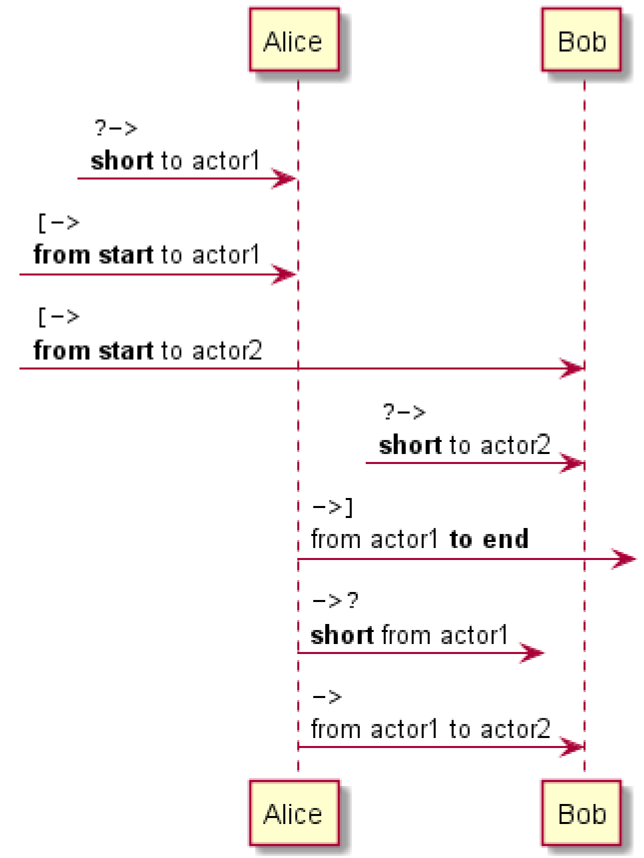
```
@startuml

?-> Alice : ""?->""\n**short** to actor1
[-> Alice : ""[->""\n**from start** to actor1
[-> Bob : ""[->""\n**from start** to actor2
?-> Bob : ""?->""\n**short** to actor2

Alice ->] : ""->]""\nfrom actor1 **to end**
```

```
Alice ->? : ""->?""\n**short** from actor1
Alice -> Bob : ""->"" \nfrom actor1 to actor2

@enduml
```



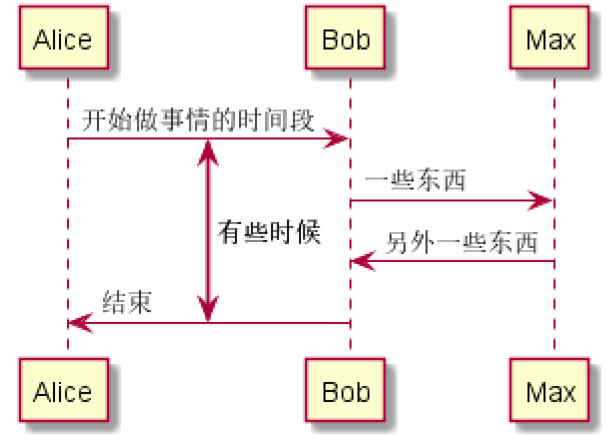
1.1.3.9 锚点和持续时间

使用 teoz 可以向图表添加锚点,并使用锚点来指定持续时间.

```
@startuml
!pragma teoz true

{start} Alice -> Bob : 开始做事情的时间段
Bob -> Max : 一些东西
Max -> Bob : 另外一些东西
{end} Bob -> Alice : 结束

{start} <-> {end} : 有些时候
@enduml
```

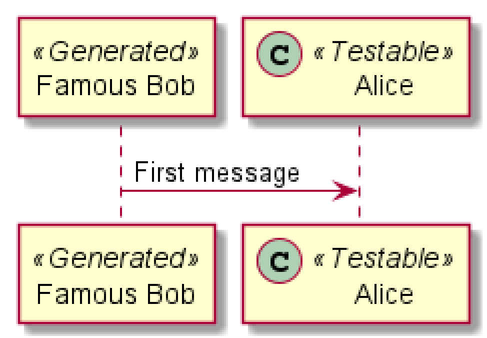


1.1.3.18 立体类型和斑点

可以使用<<和>>向参与者添加立体类型. 在定型中,可以使用语法 `(X,color)`在彩色圆圈中添加一个斑点字符.

```
@startuml
participant "Famous Bob" as Bob << Generated >>
participant Alice << (C,#ADD1B2) Testable >>

Bob->Alice: First message
@enduml
```



默认情况下,guillemet 字符被用来显示定型.你可以使用skinparam guillemet 来改变这一行为.

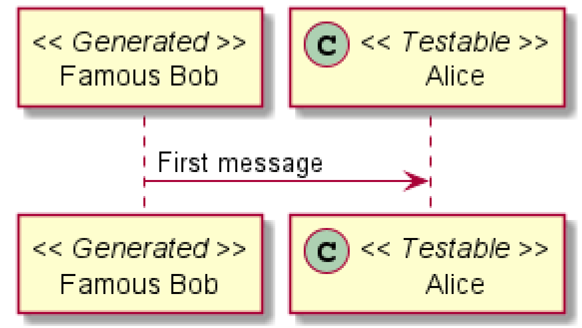
Listing 1: 也就是书名括号的形状有变化

```
@startuml
skinparam guillemet false

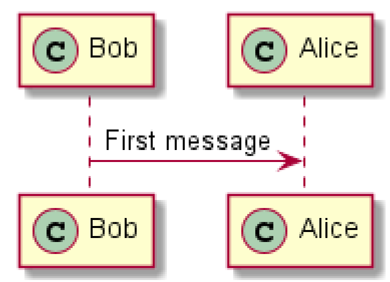
participant "Famous Bob" as Bob << Generated >>
participant Alice << (C,#ADD1B2) Testable >>

Bob->Alice: First message
```

```
@enduml
```




```
@startuml
participant Bob << (C,#ADD1B2) >>
participant Alice << (C,#ADD1B2) >>
Bob->Alice: First message
@enduml
```

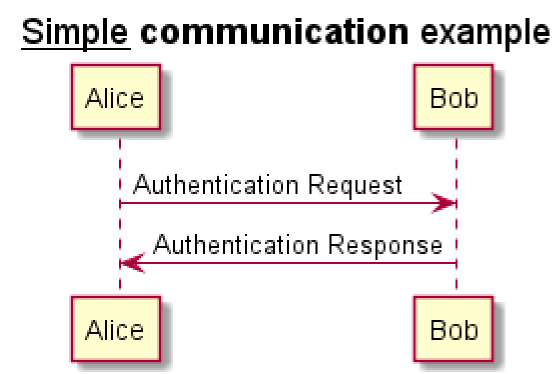


1.1.3.11 有关标题的更多信息 [可以添加图片!]

可以在标题中使用 creole 格式.

```
@startuml
title __Simple__ **communication** example

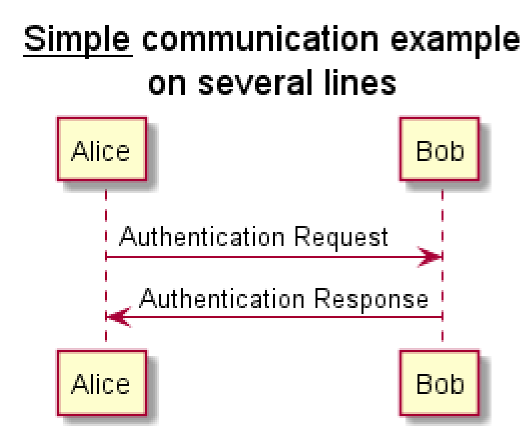
Alice -> Bob: Authentication Request
Bob -> Alice: Authentication Response
@enduml
```



1.你可以在标题描述中添加换行符

```
@startuml
title __Simple__ communication example\non several lines

Alice -> Bob: Authentication Request
Bob -> Alice: Authentication Response
@enduml
```



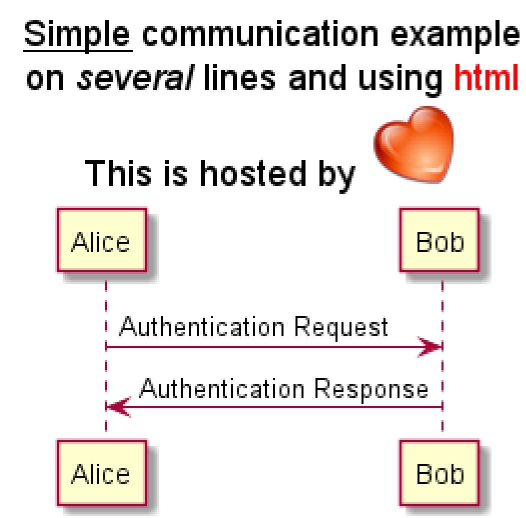
2.你还可以使用 title 和 end title 关键字在多行上定义标题.

```
@startuml

title
<u>Simple</u> communication example
on <i>several</i> lines and using <font color=red>html</font>
This is hosted by <img:icon.png>
end title

Alice -> Bob: Authentication Request
Bob -> Alice: Authentication Response
```

```
@enduml
```



1.1.3.12 参与者环绕方框

可以使用box和end box命令,在一些参与者周围画一个方框.

可以在方框关键词之后添加一个可选的标题或可选的背景颜色.

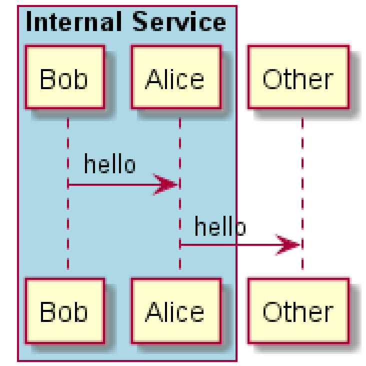
```
@startuml

box "Internal Service" #LightBlue
participant Bob
participant Alice
end box

participant Other

Bob -> Alice : hello
Alice -> Other : hello

@enduml
```

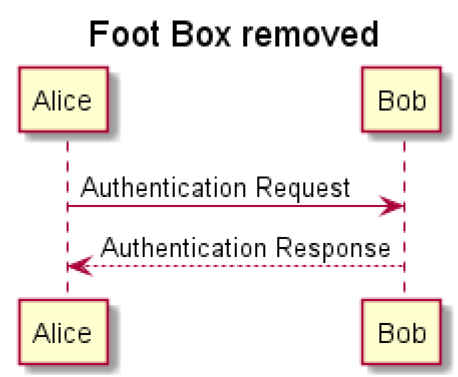


1.1.3.13 移除 Foot Boxes

你可以使用hide footbox关键字来移除图表中的脚注框.

```
@startuml
title Foot Box removed

Alice -> Bob: Authentication Request
Bob --> Alice: Authentication Response
@enduml
```



1.1.3.14 皮肤参数

你可以使用skinparam命令来改变绘图的颜色和字体.

- 在图的定义中
- 在包含的文件中
- 在一个配置文件中,在命令行或ANT任务中提供.

也可以改变其他渲染参数,如下所示:

```
@startuml
skinparam sequenceArrowThickness 2
skinparam roundcorner 20
skinparam maxmessagesize 60
skinparam sequenceParticipant underline

actor User
participant "First Class" as A
participant "Second Class" as B
participant "Last Class" as C

User -> A: DoWork
activate A

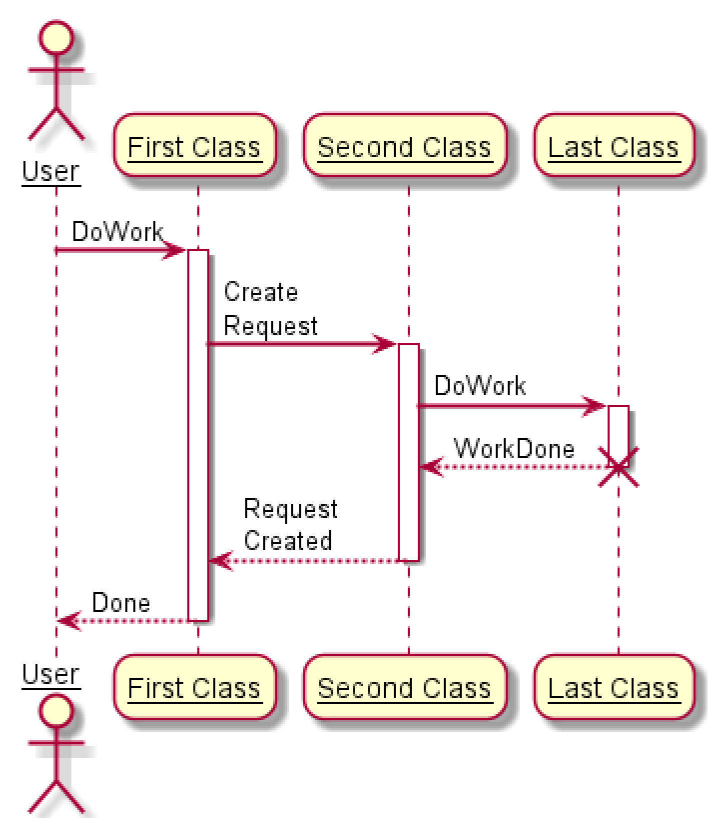
A -> B: Create Request
activate B

B -> C: DoWork
activate C

C --> B: WorkDone
destroy C

B --> A: Request Created
deactivate B

A --> User: Done
deactivate A
@enduml
```



1.1.3.14.1 案例二

```
@startuml
skinparam backgroundColor #EEEBDC
skinparam handwritten true

skinparam sequence {
        ArrowColor DeepSkyBlue
        ActorBorderColor DeepSkyBlue
        LifeLineBorderColor blue
        LifeLineBackgroundColor #A9DCDF
        ParticipantBorderColor DeepSkyBlue
        ParticipantBackgroundColor DodgerBlue
        ParticipantFontName Impact
        ParticipantFontSize 17
        ParticipantFontColor #A9DCDF
        ActorBackgroundColor aqua
        ActorFontColor DeepSkyBlue
        ActorFontSize 17
        ActorFontName Aapex
}

actor User
participant "First Class" as A
participant "Second Class" as B
participant "Last Class" as C
User -> A: DoWork

activate A
A -> B: Create Request

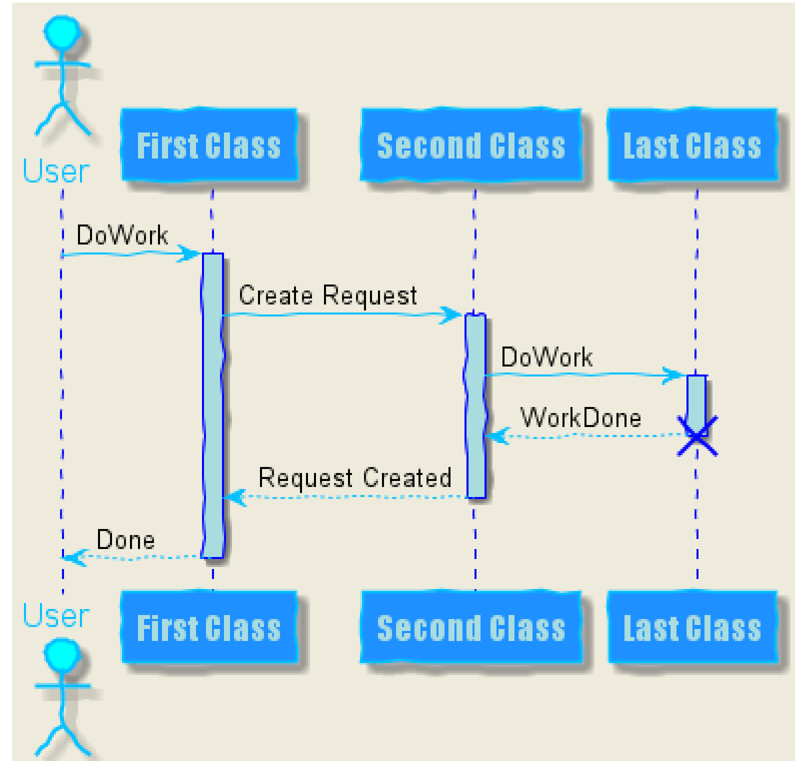
activate B
B -> C: DoWork

activate C
C --> B: WorkDone
destroy C

B --> A: Request Created
deactivate B

A --> User: Done
deactivate A

@enduml
```



1.1.3.15 改变内间距

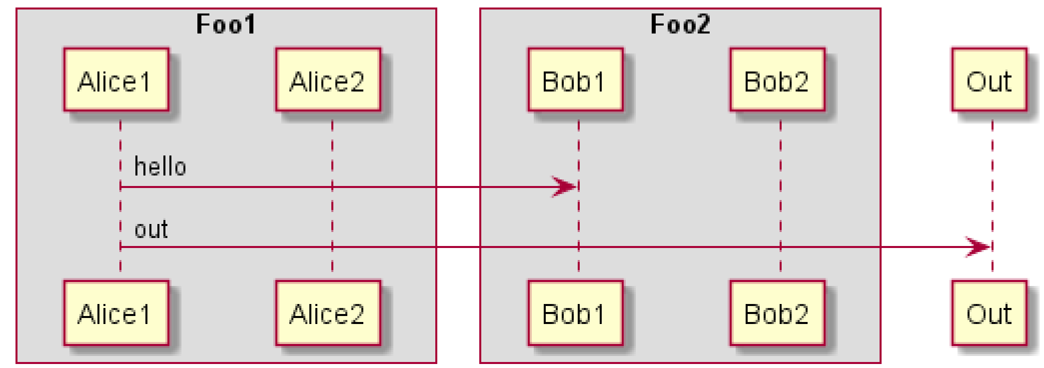
```
@startuml

skinparam ParticipantPadding 20
skinparam BoxPadding 10

box "Foo1"
participant Alice1
participant Alice2
end box

box "Foo2"
participant Bob1
participant Bob2
end box

Alice1 -> Bob1 : hello
Alice1 -> Out : out
@enduml
```



1.1.4 各种箭头

1.1.4.1 正常的箭头

```
@startuml
participant Alice as a
participant Bob as b

a -> b : ""-> ""

a ->> b : ""->> ""

a -\ b : ""-\ ""

a -\\ b : ""-\\\\""

a -/ b : ""-/ ""

a -// b : ""-// ""

a ->x b : ""->x ""

a x-> b : ""x-> ""

a o-> b : ""o-> ""

a ->o b : ""->o ""

a o->o b : ""o->o ""

a <-> b : ""<-> ""

a o<->o b : ""o<->o""

a x<->x b : ""x<->x""

a ->>o b : ""->>o ""

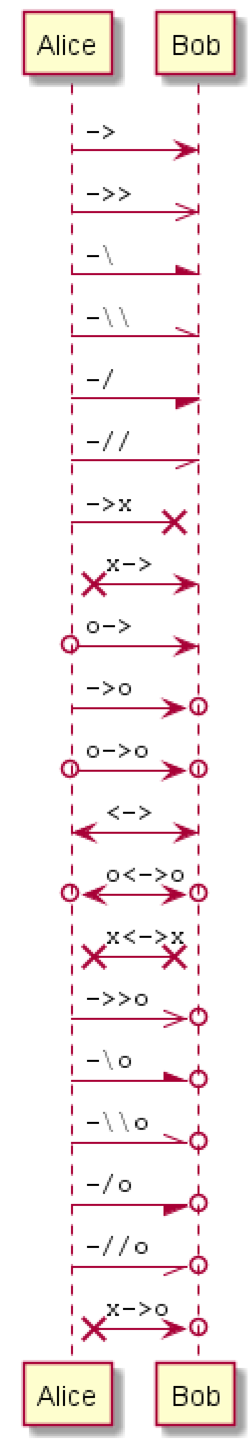
a -\o b : ""-\o ""
```

```
a -\\o b : ""-\\\\o""

a -/o b : ""-/o ""

a -//o b : ""-//o ""

a x→o b : ""x→o ""
@enduml
```



1.1.4.2 收到的信息 (带 [)

```
@startuml

participant Alice as a
participant Bob as b

[-> b : ""[-> ""

[->> b : ""[->> ""

[-\ b : ""[-\ ""

[-\\ b : ""[-\\\\""

[-/ b : ""[-/ ""

[-// b : ""[-// ""

[->x b : ""[->x ""

[x-> b : ""[x-> ""

[o-> b : ""[o-> ""

[->o b : ""[->o ""

[o->o b : ""[o->o ""

[<-> b : ""[<-> ""

[o<->o b : ""[o<->o""

[x<->x b : ""[x<->x""

[->>o b : ""[->>o ""

[-\o b : ""[-\o ""

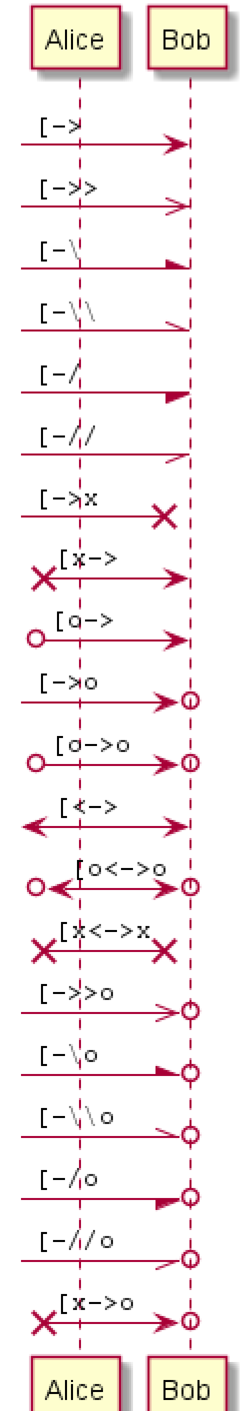
[-\\o b : ""[-\\\\o""

[-/o b : ""[-/o ""

[-//o b : ""[-//o ""

[x->o b : ""[x->o ""

@enduml
```



1.1.4.3 发出的信息 [带]]

```
@startuml
participant Alice as a
participant Bob as b

a ->] : ""->] ""
```

```
a →>] : ""→>] ""

a -\] : ""-\] ""

a -\\] : ""-\\\\]""

a -/] : ""-/] ""

a -//] : ""-//] ""

a →x] : ""→x] ""

a x→] : ""x→] ""

a o->] : ""o->] ""

a →o] : ""→o] ""

a o->o] : ""o->o] ""

a ←>] : ""←>] ""

a o←>o] : ""o←>o]""

a x←>x] : ""x←>x]""

a →>o] : ""→>o] ""

a -\o] : ""-\o] ""

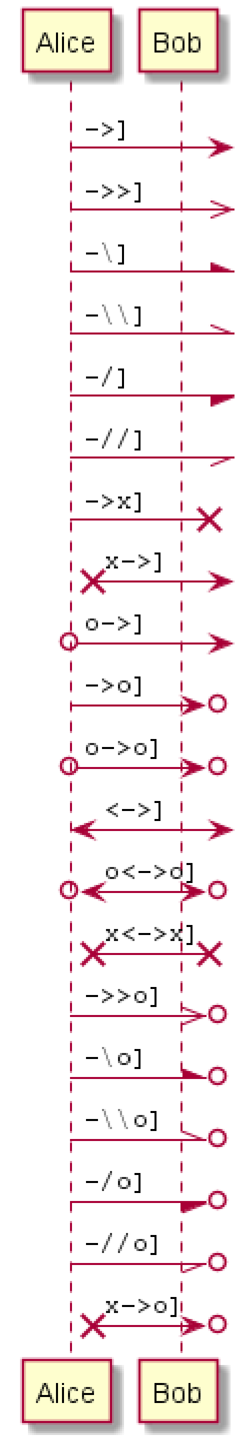
a -\\o] : ""-\\\\o]""

a -/o] : ""-/o] ""

a -//o] : ""-//o] ""

a x→o] : ""x→o] ""

@enduml
```



1.1.4.4 简短的传入信息（带"?"）

```
@startuml
participant Alice as a
participant Bob as b
a -> b : //Long long label//

?-> b : ""?-> ""
```

```
?->> b : ""?->> ""

?-\ b : ""?-\ ""

?-\\ b : ""?-\\\\""

?-/ b : ""?-/ ""

?-// b : ""?-// ""

?->x b : ""?->x ""

?x-> b : ""?x-> ""

?o-> b : ""?o-> ""

?->o b : ""?->o ""

?o->o b : ""?o->o ""

?<-> b : ""?<-> ""

?o<->o b : ""?o<->o""

?x<->x b : ""?x<->x""

?->>o b : ""?->>o ""

?-\o b : ""?-\o ""

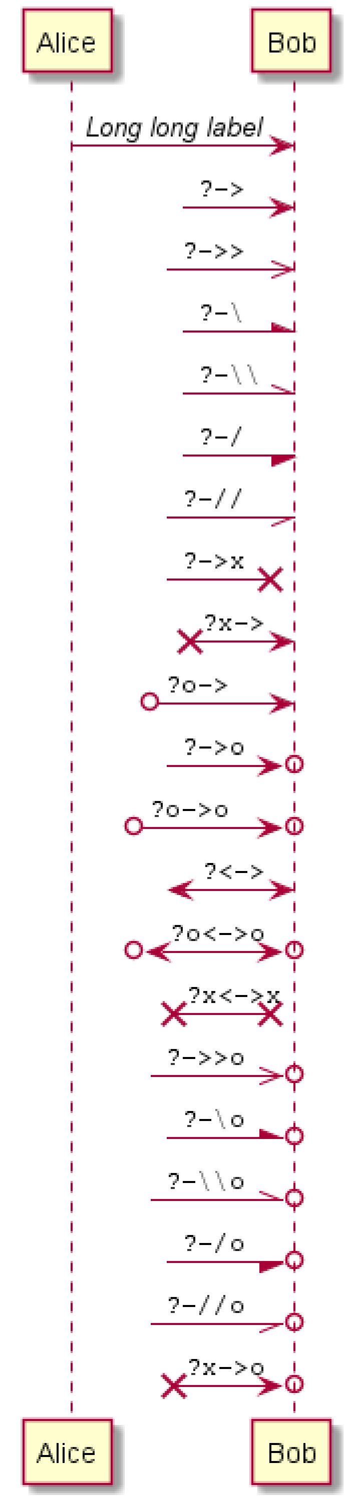
?-\\o b : ""?-\\\\o ""

?-/o b : ""?-/o ""

?-//o b : ""?-//o ""

?x->o b : ""?x->o ""

@enduml
```



1.1.4.5 简短的发出信息（带"?"）

```
@startuml

participant Alice as a
participant Bob as b
```

```
a → b : //Long long label//

a →? : ""→? ""

a →>? : ""→>? ""

a -\? : ""-\? ""

a -\\? : ""-\\\\?""

a -/? : ""-/? ""

a -//? : ""-//? ""

a →x? : ""→x? ""

a x→? : ""x→? ""

a o->? : ""o->? ""

a →o? : ""→o? ""

a o->o? : ""o->o? ""

a ←>? : ""←>? ""

a o←>o? : ""o←>o?""

a x←>x? : ""x←>x?""

a →>o? : ""→>o? ""

a -\o? : ""-\o? ""

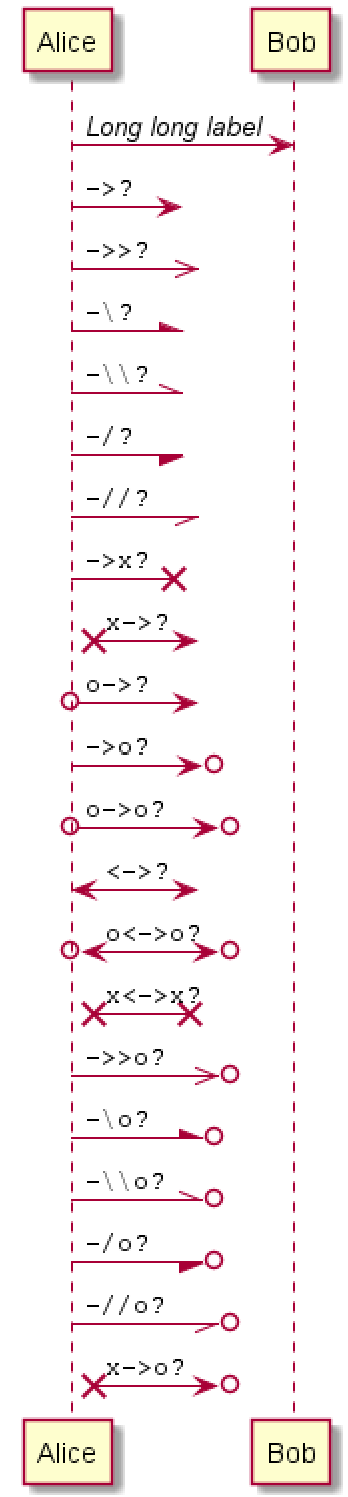
a -\\o? : ""-\\\\o?""

a -/o? : ""-/o? ""

a -//o? : ""-//o? ""

a x→o? : ""x→o? ""

@enduml
```

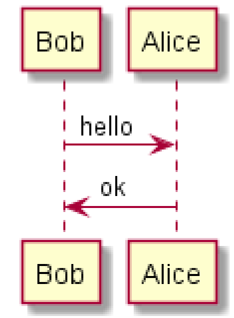


1.1.5 皮肤

1.1.5.1 默认皮肤

```
@startuml
Bob -> Alice : hello
```

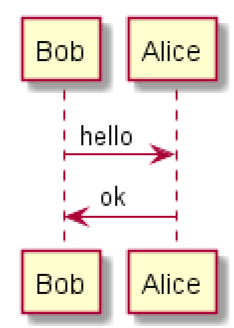
```
Alice -> Bob : ok
@enduml
```



1.1.5.2 生命线 strategy

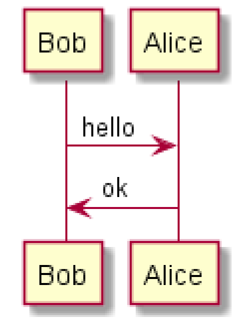
- 默认非实线 no-solid

```
@startuml
skinparam lifelineStrategy nosolid
Bob -> Alice : hello
Alice -> Bob : ok
@enduml
```



- solid

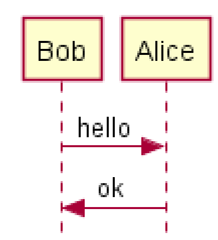
```
@startuml
skinparam lifelineStrategy solid
Bob -> Alice : hello
Alice -> Bob : ok
@enduml
```



1.1.5.3 严格控制样式

为了符合严格的UML(对于箭头样式：发出三角形而不是尖锐的箭头),你可以使用:

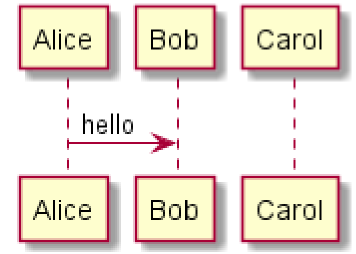
```
@startuml
skinparam style strictuml
Bob -> Alice : hello
Alice -> Bob : ok
@enduml
```



1.1.5.4 隐藏未链接的参与者

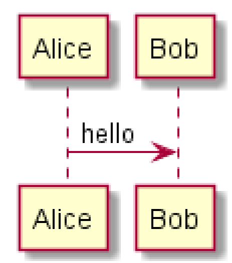
默认情况下,会展示所有参与者.

```
@startuml
participant Alice
participant Bob
participant Carol
Alice -> Bob : hello
@enduml
```



但是你可以隐藏未链接的参与者.

```
@startuml
hide unlinked
participant Alice
participant Bob
participant Carol
Alice -> Bob : hello
@enduml
```



1.1.6 FAQ

1.1.6.1 uml 中的生命线

生命线表示参与交互的单个对象

Date: 2021-09-23 Thu 22:04
Created: 2021-09-24 Fri 23:35
[Validate]